| | |
|---|---|
| **Course code and name** | GE1354 – Introduction to Electronic Design |
| **Group** | L_2_ Gp _17_ |
| **Members (Name and SID)** | 1. Chan Ka Hung SID: 55240293<br>2.Cheng Kwan Yuen SID: 55226599<br>3. Yeung Tung Yan SID: 55213094 |
| **Final Project**<br>(Health monitoring system or<br>Bike Rider Facilitating System) | Bike Rider Facilitating System |

**Section 1 - Summary (about 100 words)**

By considering the safety issue of the bike rider facilitating system, we have designed the following functions for the micro:bit.

The project involves two main parts, which are the basic functions part and the advanced functions part. For the basic functions, on/off button, signals for turn left/right/stop/slow down/hazard and button for front light are included. For the advanced functions, digital point north compass, falling alert and velocity calculation by wheel are included.

The above functions have made use of four micro:bits and the built-in sensors of micro:bit, which are magnetometer and accelerometer.

**Section 2 – Introduction (about 300 words)**

The followings are the objectives of designing each basic function and advanced function.

The **basic functions** include:

- **On/off button**, which indicates whether the system is working or not.
- Signals for **turn left/right/stop/slow down/hazard**, which lights up the back tail lights to alert the drivers behind by hand motion detection.
- **Front light**, which allows user drive in the dark environment with lighting up the front light by pressing a button.

The **advanced functions** include:

- **Point north compass**, which orientes the direction to north. Due to the reason that people rely on their GPS function on phone all the time. We consider the situation what if the phone runs out of power. The power of phone cannot stand all day long. Therefore, we have the idea of making a compass. Users may not only ride their bikes in city. Even if they are alone in the wilderness or far away from a town, as long as they have a compass, they can simply navigate across the environment without requiring the

aid of another person and without needing a phone. At least, the digital compass on micro:bit keeps them do not get lost in some emergency situations.

- **Falling alert**, which  The micro:bit on helmet would detect when the user falls, and then all the LEDs at the front tail lights would start blinking to alert the drivers behind and the speaker will play loud “beep” sound to alert people around.We have the idea of this because of the safety issue. Consider this situation, a bike falls suddenly, a big trunk is coming. Usually, driver on the big truck may not notice small bike on road. The driver may not respond in time to stop the truck. In order to lower the accident likes this, we designed this function.
- **Velocity calculation**, which calculates velocity by algorithm and show on micro:bit on right hand. We have the idea of making this because some professional riders usually train their riding speed. They can monitor their velocity all the time. If the velocity displays on their arm instantly, this function would bring advantage to their training.

**Section 3 – Objectives (in point form)**

- Providing information to the biker (such as direction, current speed)
- Indicating other road users by the back tail light
- Alerting others when accident happens

**Section 4 – Background Theory (about 500 words)**

Magnetometer is a built-in sensor, which is used to measure the Earth's magnetic field in each of the three perpendicular axes, X, Y and Z. There is a sensor which is going to give us the readings and based on those reading we are going to decide the directions. The driver of the magnetometer returns raw values. Each magnetometer is different, and require calibration to account for raw numbers offsets and magnetic field distortions introduced by what are known as soft and hard iron interference.

We used this sensor to create a digital compass. We first figure out the bearing of the compass by dividing it into eight portions. And then, we start the programming. We assigned eight directions, north, east, south, west, northeast, northwest, southeast, and southwest. Each direction was assigned by 45 degrees.

Accelerometer is a sensor that measures the gravitational forces pulling on it in all, which is the acceleration (in milligram) along three dimensions of the chips: x and y (horizontal plane), and z (the vertical plane).

We used this sensor as gesture detection. Because we want the micro:bit to react to changing circumstance, we use a forever loop. Within the scope of the loop the gesture with assigned acceleration is read and output signal to turn on LEDs.

We also used this sensor as falling detect. Once again, because we want the device to react to changing circumstances, we use a forever loop. Within the scope of the loop, the current acceleration of helmet is read. The if condition checks if accelerations of x

or y is smaller than -900. If the acceleration is less than -900, then play beep sound and light up all the back tail lights. Otherwise, nothing would be triggered.

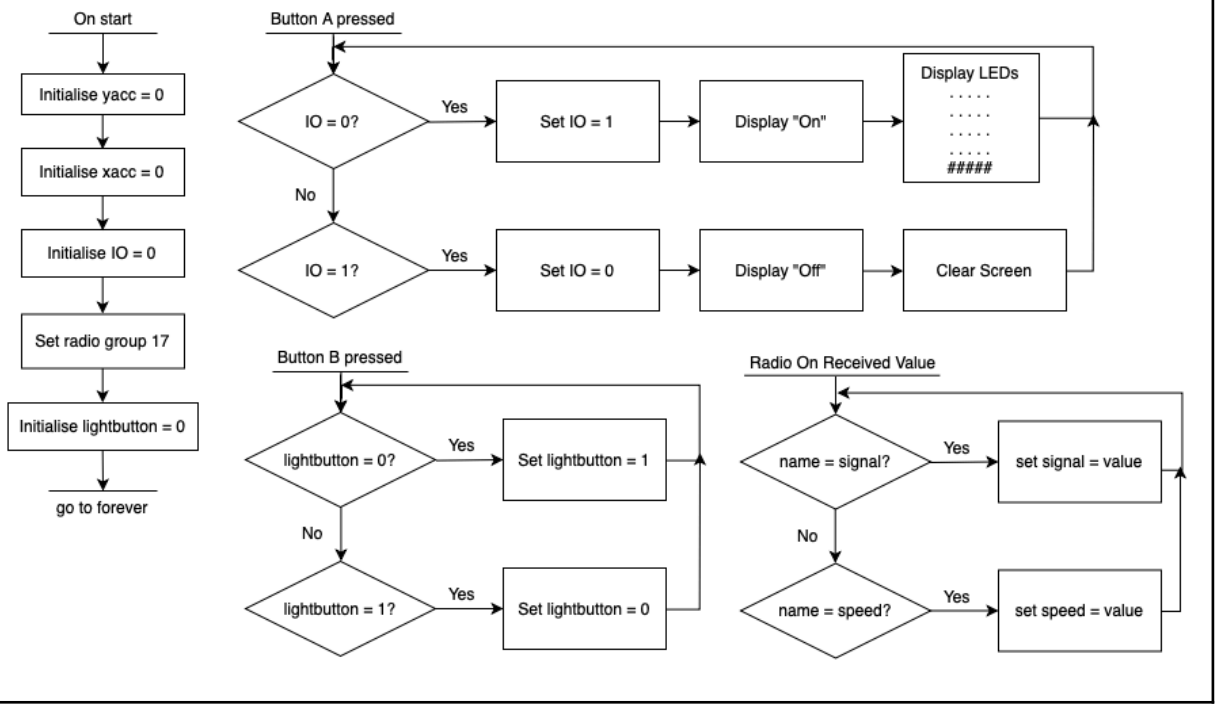
We used this sensor as velocity calculation as well. With using the concept of centripetal force, under rotation circumstances , force is given with vector pointing toward the rotation center parallel to the rotation plane. To conduct the concept into a working speedometer , we need to install an accelerometer on the side of the wheel with a certain distance to the center rotation , the centroid  of the wheel , which is the radius . for a more accurate measurement , the radius should be as large in length as possible. Since the acceleration is in fact the measurement of change of speed over change of time , we need to make sure the duration between each measurement , simply by setting pause between each measurement and input the pause delay as the reference of time difference will gives a sector of period of time with respect to the changes in speed , thought this , we are able to squeeze out the current operating speed from the formula.

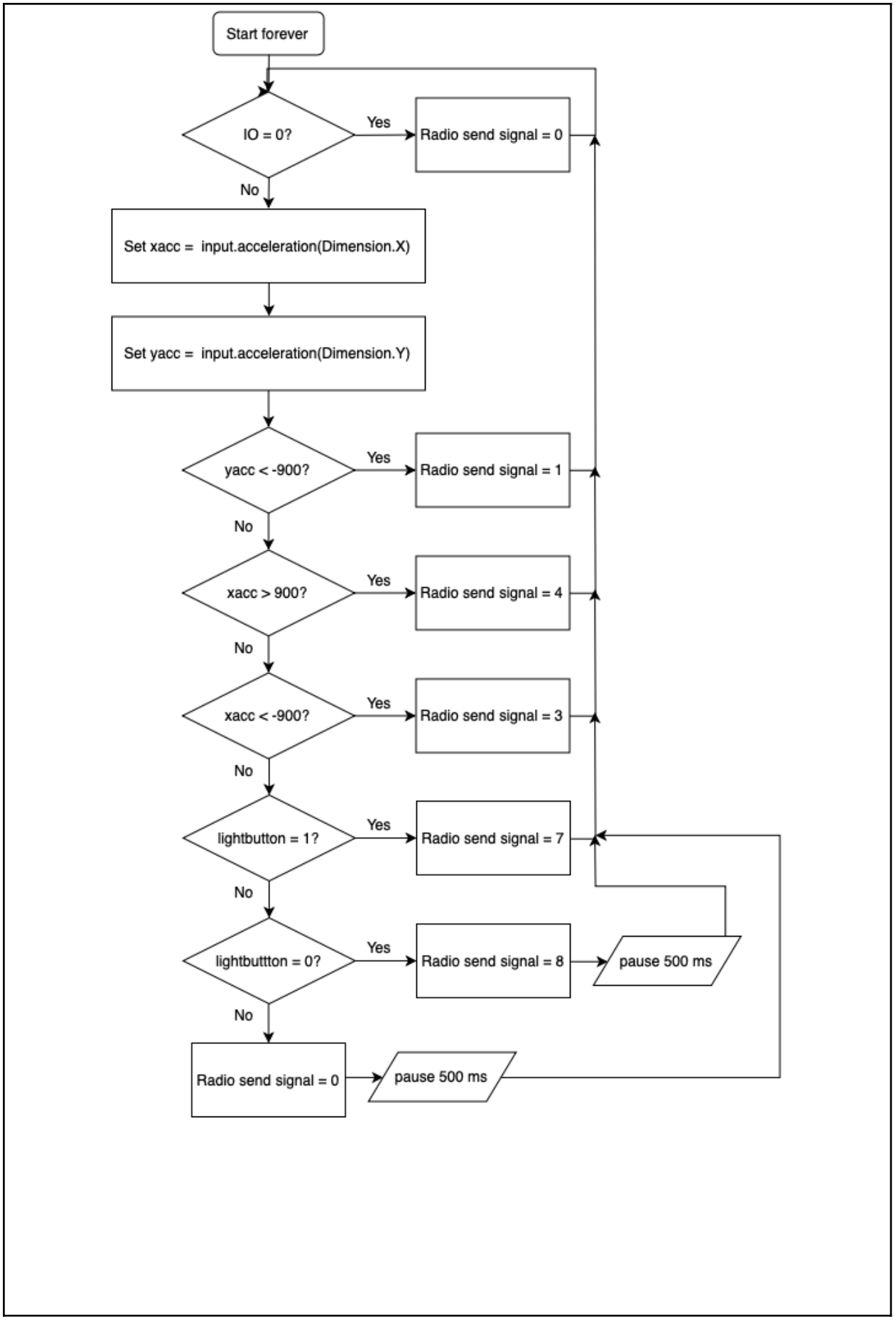
Radio is a built-in component for sending and receiving data using wireless interaction. The smallest useful part of the signal of radio is a byte. A byte is an unit of information that consists of eight bits. A bit is the smallest possible unit of information as it could only be in two states: ON or OFF. By bytes, we can represent numbers between 0 and 255, which represents character, then we can send one character per byte at a time through air. We can't transmit directly to one person with radio. Anyone with an appropriate aerial can receive the messages you send out. As a result, we need to first set radio group for who is receiving broadcasts.

We used radio signal throughout the whole program. By the event of biker fell on ground with helmet, the micro:bit  of helmet sends radio value, event blocks on another micro:bit will be triggered any time a number is received, over the radio on group 17.

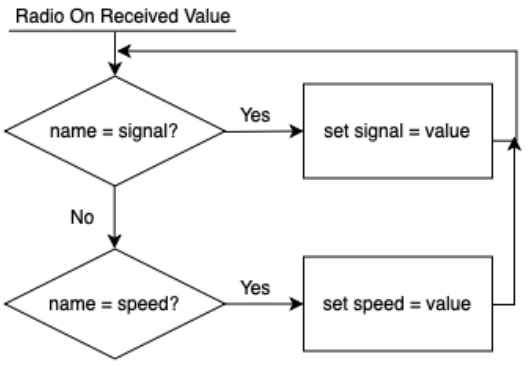
**Section 5 – Hardware and Software Design (about 500 words)**

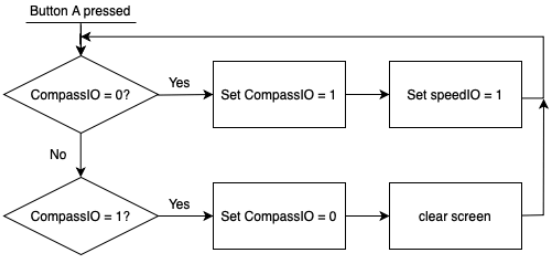
Left hand:

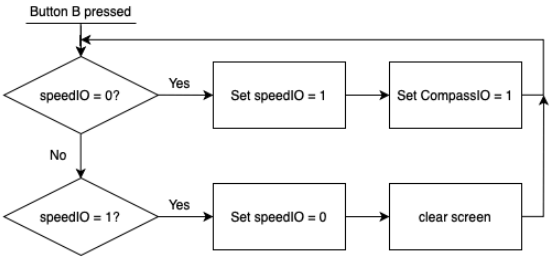

On start
Initialise yacc = 0
Initialise xacc = 0
Initialise IO = 0
Set radio group 17
Initialise lightbutton = 0
go to forever
Button A pressed
IO = 0?
Yes
Set IO = 1
Display "On"
Display LEDs
. . . . .
. . . . .
. . . . .
. . . . .
#####
No
IO = 1?
Yes
Set IO = 0
Display "Off"
Clear Screen
Button B pressed
lightbutton = 0?
Yes
Set lightbutton = 1
No
lightbutton = 1?
Yes
Set lightbutton = 0
Radio On Received Value
name = signal?
Yes
set signal = value
No
name = speed?
Yes
set speed = value

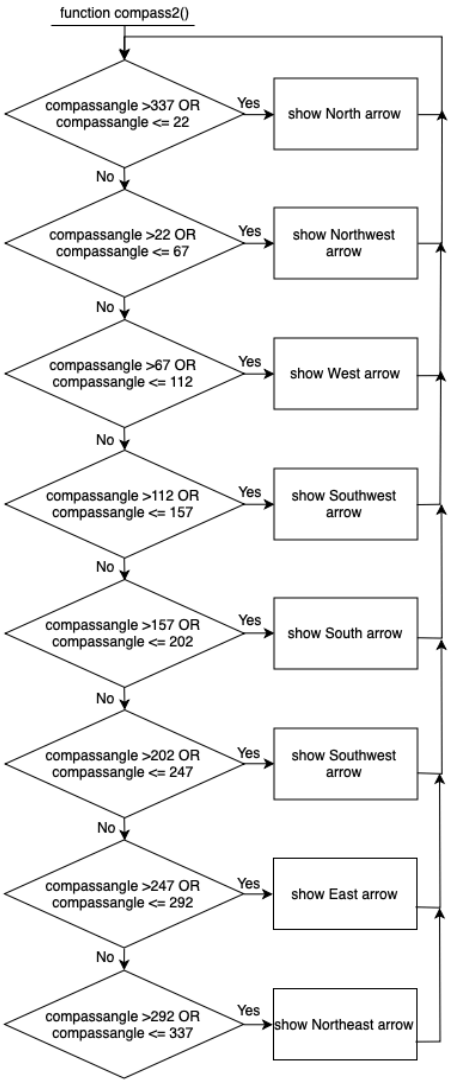

Start forever
IO = 0?
Yes
Radio send signal = 0
No
Set xacc = input.acceleration(Dimension.X)
Set yacc = input.acceleration(Dimension.Y)
yacc < -900?
Yes
Radio send signal = 1
No
xacc > 900?
Yes
Radio send signal = 4
No
xacc < -900?
Yes
Radio send signal = 3
No
lightbutton = 1?
Yes
Radio send signal = 7
No
lightbuttton = 0?
Yes
Radio send signal = 8
pause 500 ms
No
Radio send signal = 0
pause 500 ms

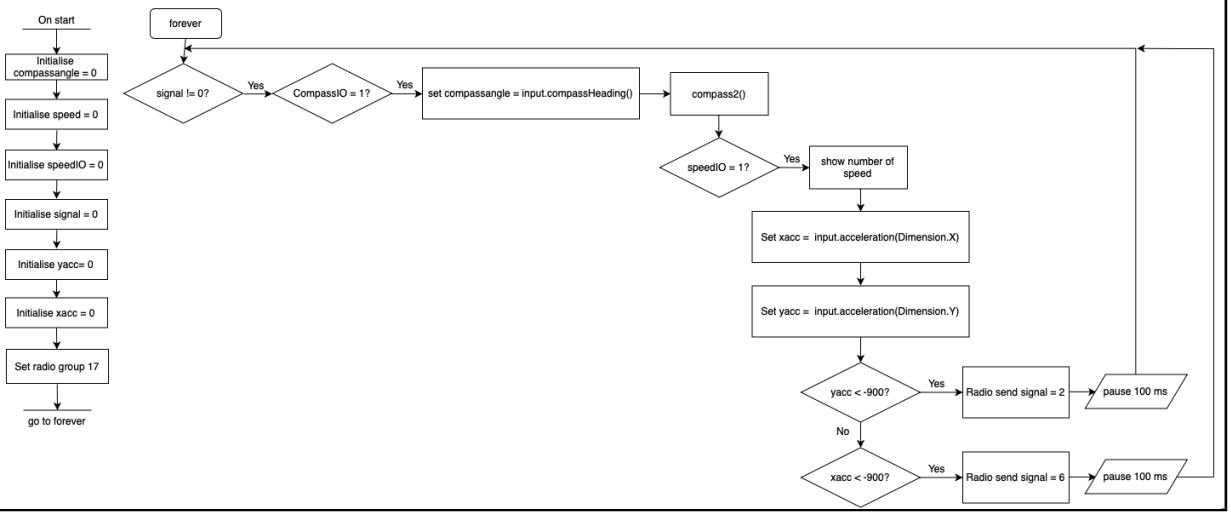
Right hand:


Radio On Received Value
name = signal?
Yes
set signal = value
No
name = speed?
Yes
set speed = value

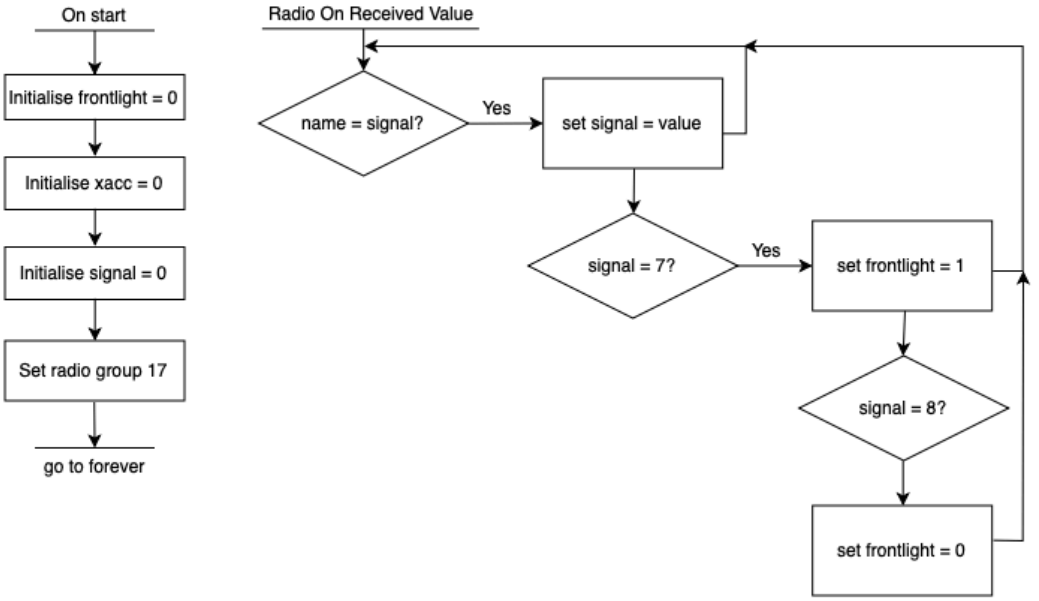

Button A pressed
CompassIO = 0?
Yes
Set CompassIO = 1
Set speedIO = 1
No
CompassIO = 1?
Yes
Set CompassIO = 0
clear screen

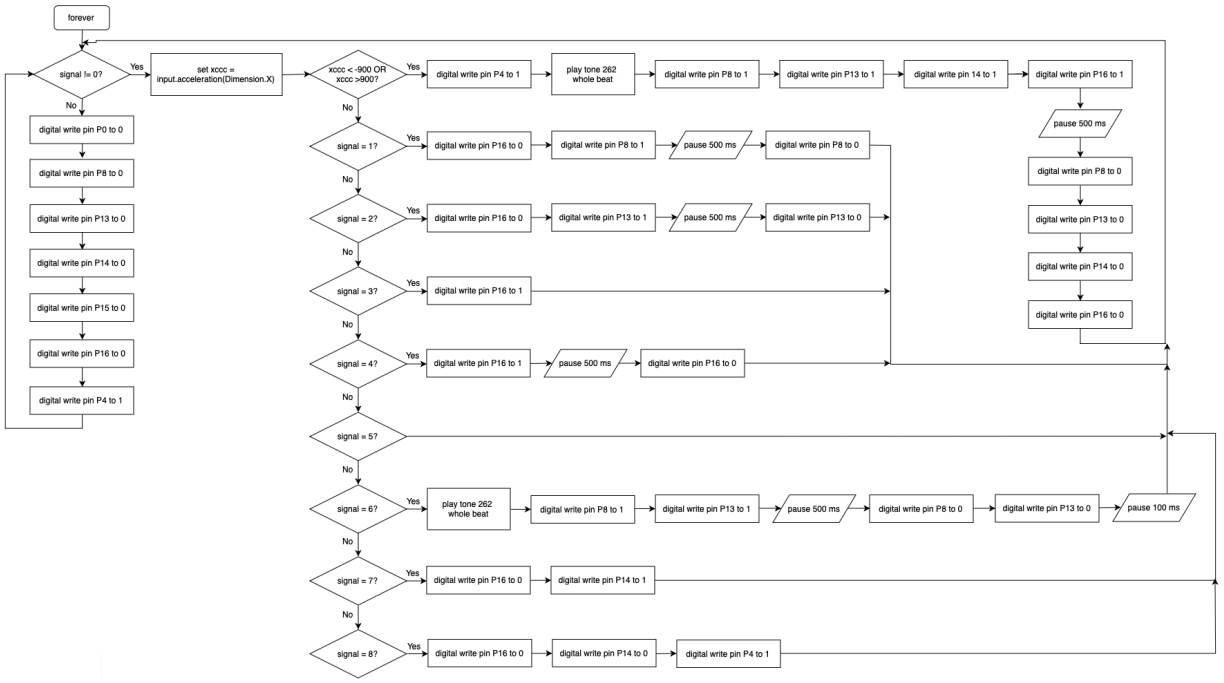

Button B pressed
speedIO = 0?
Yes
Set speedIO = 1
Set CompassIO = 1
No
speedIO = 1?
Yes
Set speedIO = 0
clear screen


function compass2()
compassangle >337 OR compassangle <= 22
Yes
show North arrow
No
compassangle >22 OR compassangle <= 67
Yes
show Northwest arrow
No
compassangle >67 OR compassangle <= 112
Yes
show West arrow
No
compassangle >112 OR compassangle <= 157
Yes
show Southwest arrow
No
compassangle >157 OR compassangle <= 202
Yes
show South arrow
No
compassangle >202 OR compassangle <= 247
Yes
show Southwest arrow
No
compassangle >247 OR compassangle <= 292
Yes
show East arrow
No
compassangle >292 OR compassangle <= 337
Yes
show Northeast arrow

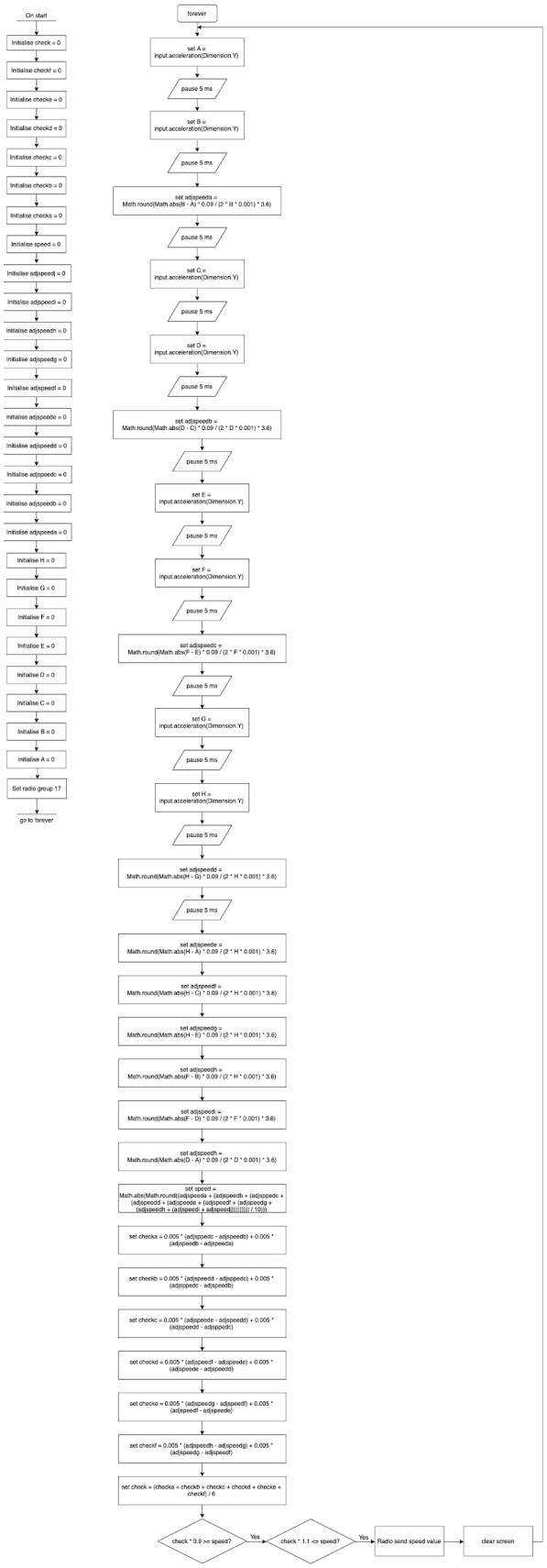

On start
Initialise compassangle = 0
Initialise speed = 0
Initialise speedIO = 0
Initialise signal = 0
Initialise yacc= 0
Initialise xacc = 0
Set radio group 17
go to forever
forever
signal != 0?
Yes
CompassIO = 1?
Yes
set compassangle = input.compassHeading()
compass2()
speedIO = 1?
Yes
show number of speed
Set xacc = input.acceleration(Dimension.X)
Set yacc = input.acceleration(Dimension.Y)
yacc < -900?
Yes
Radio send signal = 2
pause 100 ms
No
xacc < -900?
Yes
Radio send signal = 6
pause 100 ms

Helmet:

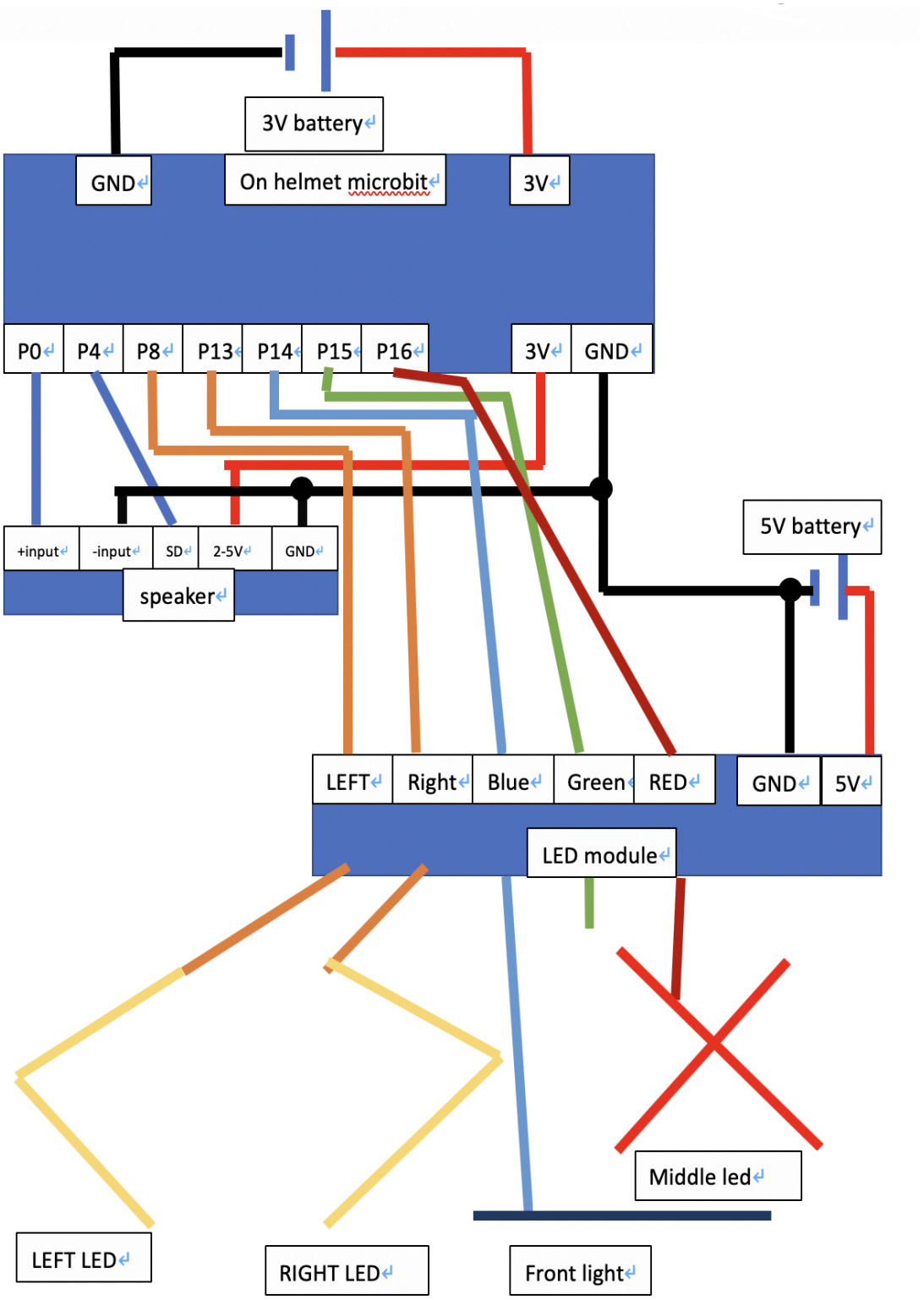

On start
Initialise frontlight = 0
Initialise xacc = 0
Initialise signal = 0
Set radio group 17
go to forever
Radio On Received Value
name = signal?
Yes
set signal = value
signal = 7?
Yes
set frontlight = 1
signal = 8?
set frontlight = 0


forever
signal != 0?
Yes
No
set xccc = input.acceleration(Dimension.X)
digital write pin P0 to 0
digital write pin P8 to 0
digital write pin P13 to 0
digital write pin P14 to 0
digital write pin P15 to 0
digital write pin P16 to 0
digital write pin P4 to 1
xccc < -900 OR xccc >900?
digital write pin P4 to 1
play tone 262 whole beat
digital write pin P8 to 1
digital write pin P13 to 1
digital write pin 14 to 1
digital write pin P16 to 1
pause 500 ms
digital write pin P8 to 0
digital write pin P13 to 0
digital write pin P14 to 0
digital write pin P16 to 0
signal = 1?
digital write pin P16 to 0
digital write pin P8 to 1
pause 500 ms
digital write pin P8 to 0
signal = 2?
digital write pin P16 to 0
digital write pin P13 to 1
pause 500 ms
digital write pin P13 to 0
signal = 3?
digital write pin P16 to 1
signal = 4?
digital write pin P16 to 1
pause 500 ms
digital write pin P16 to 0
signal = 5?
signal = 6?
play tone 262 whole beat
digital write pin P8 to 1
digital write pin P13 to 1
pause 500 ms
digital write pin P8 to 0
digital write pin P13 to 0
pause 100 ms
signal = 7?
digital write pin P16 to 0
digital write pin P14 to 1
signal = 8?
digital write pin P16 to 0
digital write pin P14 to 0
digital write pin P4 to 1

Wheel: (zoom in and check)



Hardware connection:



For the **Hardware component**, we mainly have 4 micro:bits using wireless connection and one of the micro:bit will make use of edge connector to connect to the speaker and LED.

Firstly, we used 4 micro:bits, one for the left hand gesture detection; One for the right hand gesture detection; One for installing on the wheel to measure the speed, and one install on helmet to receive signal and output the voltage to the LED and speakers. For these 4 micro:bits that we use, we all using radio communication to transfer the data.

Secondly, we use wire connection to connect the on helmet micro-bit to the speaker and LED circuit.

For the speaker, we need 5 pins to control the speaker. Therefore, we assign 5 pin from the micro-bit edge connector to do this job. Pin 0 will be connected to the +Input pin on the speaker module, so that the sound signal can send to the speaker. -Input port would connect to the GND of the microbit, as we do not have -input. Shutdown signal will connect to the pin4. 2-5V pin will connect to the 3V output from the micro-bit. GND will connect to the 0V of the micro-bit.

For the LED, we need to assign 5 signal pin from the micro-bit to the LED. Since the LED need a high power, we provide the LED with 5V, and the ground from the LED should also connect to the ground of the battery. One thing to notice is that the GND pin from the 5V battery should also connect to the micro:bit edge connector's 0V, in order to ground it. For the signal pin, we assigned pin 8 as left LED, pin 13 as right LED, pin 14 as blue LED, pin 15 as green LED, pin 16 as red LED signal.

For the **software design**, we have 3 major area.

First, we designed a On/Off function. Since mirco:bits do not have a formal button to turn on or off. Once we plug in the battery, the micro:bits turn on automatically. Therefore, the users are difficult to determine whether the micro:bits are working or not. In our design, when the Button A of the Left-hand micro:bits are pressed, the system will be turned on and a string "ON" will be shown on the LED. When button A pressed again, a string "OFF" will be shown on the LED to indicate that the LED is off.

Second, input to the micro-bit and send the signal. In this project, we mainly have 3 kinds of input. First, we use the gesture to determine which signal to be sent. Take turn left as an example, we designed that showing a left straight hand to indicate to turn left. To this gesture, the microbit will face down. To be more accurate, the accelerometers' Y-axis would have a number close to -1023. And we use y<-900 as a standard range to

determine this action. Then these gesture will turn into a signal and use radio signal send to the on helmet microbit. Second kind we would have button interrupt, such as left-hand microbit's button B. When this is pressed, we send signal to the helmet microbit and turn on the front light. Third, since we use a micro-bit and installed on a wheel. This on wheel micro:bit are used to calculate the speed of the wheel and send the signal to the right hand micro:bit. When right hand micro:bit's button B are pressed, the speed will should one the LED.

Third, output according to the input signal. Most of the signal will be send to the on helmet micro-bit decide which light, or what sound output. Taking left turn as an example again. when left hand microbit detected the motion, it will send a signal "1". When the on helmet microbit received this signal, it will output a digital '1' to the Pin8. As mentioned in the hardware part, we connected our pin8 to the left light. As a result, left light will lighted up to indicate a left turn.

**Section 6 – Principle of Operation**

In our bike rider facilitating system, we have 3 major objective, Providing information to the biker, indicating other road user, and alert others when accident happen.

First, providing information. In our design, there are two function that provide information to the bike rider.

First, we have a **compass**, providing the the biker direction. When button A on the right micro-bit pressed, a compass will show through the LED. This compass will always pointing to the North side so the biker can know the direction that is riding on. This function is base on the build in magnetometer to work.

Second, we have a **speed function**. When button B on the right micro-bit pressed, the speed will be show on the LED. The principle of this function is because we have installed a micro-bit on the wheel. The micro-bit will continuously calculate the change acceleration and convert to velocity and output on the right hand micro-bit LED.

Second, indicating other user. We have 5 major function is this part, including left turn, right turn, stot, slow down and front light.

First, when we show a straight left hand, this indicate **left turn**. The left light will blink.

Second, when we show a straight right hand, this indicate **right turn**, The right light will blink.

Third, when we lay down our left hand, this indicate **slow down**. The red led on the helmet will blink.

Fourth, when we raise up our left hand, this indicate **stop**. The red led on the helmet will light up until will put down our hand.

These above four function is base on the accelerometer on the microbit. When the accelerometer exceed some value, the micro-bit will send out the signal to the helmet microbit to proceed different light.

Fifth, we have a **front light** function. When button B on the left hand micro-bit pressed, the LED installed at the front of the helmet will be turned on. Since safety is the first priority when riding a bicycle. At night, the car in front is hard to notice the biker at the back. Therefore, it is necessary to turn on the front light, to call attention to the driver at the front aware there are a bike at the back.

Third, alerting others when accident happens. We have two function in this part, alerting other by pointing right hand to the ground and falling alert.

First, **alerting** other by pointing right hand to the ground. By doing this action, the LED at the bike will blink and speaker will make some some to alert others. When biker see there are some accident on the front, or when biker notice that there are some obstacle on the ground, they can do this gesture to alert others.

Second, **falling alert**. Since we have a micro-bit installed on the helmet, when the biker fall on the left or right side, the accelerometer will detect the angle. When this angle is too high, that may means that the biker fall down. As a result, all the LED will blink and speaker will make some sound to alert others. Since our system is used on bike, it is hard to notice by other when the biker have any accident. Specially for the truck driver, as they have a higher seat position, they are hard to notice the bike at the front. Therefore, it is necessary to notice others by sound rather just by vision.

**Section 7 – Analysis and Discussion**

In this microbit project, we have achieved a great goal. In our design, all the lights and sounds are generated correctly by doing different hand gesture.

For the basic function, the system actually depends on the accelerometer to work. In microbit accelerometer, it divided into 1024 scale, where it is from 0 to 1023. We choose 900 this value as our trigger point. Since we have a quite large trigger range, the system are easily trigger, and no need to be absolute degree. Therefore, I would comment that this system is easy to use.

For the advance function, we may have 3 point to discuss about, which are the magnetometer, speed, and the light sensor.

For the magnetometer, we actually use this to measure the direction, and provide the North direction to the user. However, what we find that is the build in magnetometer is not so accurate as we think. By using a real compass as reference data, we find that the microbit magnetometer have around 10 to 20 degree error. It is hard to get the absolute direction. Therefore, we divided the representation of the direction into 8 spot, such as North, North-East, East. We are trying to provide the nearest result to the user.

For the speed, we actually build a small wheel to demonstrate the uniform circular motion. However, we find that it is pretty hard. First of all, when we are building the wheel, it is difficult to make it rotate as a straight line. The wheel are easily inclined to the left or right. Moreover, after we installed the micro-bit on the wheel, this inclined phenomenon become more serious, as the weight of the micro-bit. In addition, the friction, the radius of the wheel is also a factor that affect the accurate of the measuring. In is hard to deal with these physics problem as being A electronic student. Therefore, we tend to solve this problem by using programming skills. In order to avoid some extreme value, we would like to use the average method. During measuring the speed, it is easy to get some extremely large value, which is not accurate. However, if we take the mean from a set of value, we can reduce this problem easily. Therefore, after testing our device, we can claim that our system have around 10% of error compare to the usual device.

For the light sensor, our original thought about the front light is like this: We want to automatically turn on the front light at dark environment, being at night, or inside a tunnel. However, after testing the light sensor on the micro-bit, we find out that this thought is not that possible. One of the biggest problem is that the accuracy of the light sensor. As from the data sheet, we know that the light sensor have a range between 0 to 255. However, during our measurement, the light sensor is not that stable. We need to cover our full hand on the microbit so to get value 0. Even at a bright room, it is not possible to get 255. We need to turn on the torch to get the value 255. Due to this unstable feature, it is hard to build a automatically turn on system. As a result, we have a alternative way to do this. We change this function to press the left hand button B to trigger. It is important that the system work according to the user's willingness. When an automatic- system is not possible, it is better to make it to be turn on/off by hand.

**Section 8 – Conclusion (100 words)**

The project has successfully brought out the three major initial objectives as mentioned. By applying this bike rider facilitating system into reality, it is able to provide direction and velocity to the biker, indicating other road users by the back tail lights, and alerting others when accident happens on road.

For the front light, back tail lights and falling alert are absolute accurate, while the other two parts have minor errors. The compass has 10-20 degrees of error due to the capability of the magnetometer. For the velocity calculation, by avoiding wrong data, we take multiple data and divide them into different combinations which allows we to minimize the chance of getting unreasonable output. We lower the tolerance to around 10%.

**Section 9 – References**

https://microbit-micropython.readthedocs.io/fr/stable/tutorials/radio.html

**Appendix:**

**Left hand**

```
let yacc = 0
let lightbutton= 0
let xacc = 0
let IO = 0
input.onButtonPressed(Button.A, function () {
  if (IO == 0) {
    IO = 1
    basic.showString("On")
    basic.showLeds(`
        . . . . .
        . . . . .
        . . . . .
        . . . . .
        # # # # #
        `)
  } else if (IO == 1) {
    IO = 0
    basic.showString("Off")
    basic.clearScreen()
  }
})
input.onButtonPressed(Button.B, function () {
```

```
  if (lightbutton == 0) {
      lightbutton = 1
   } else if (lightbutton == 1) {
      lightbutton = 0
   }
})
radio.setGroup(17)
IO = 0
lightbutton = 0
basic.forever(function () {
  if (IO == 0) {
      radio.sendValue("signal", 0)
   } else {
      xacc = input.acceleration(Dimension.X)
      yacc = input.acceleration(Dimension.Y)
      if (yacc < -900) {
          radio.sendValue("signal", 1)
          basic.pause(500)
      } else if (xacc > 900) {
          radio.sendValue("signal", 4)
          basic.pause(500)
      } else if (xacc < -900) {
          radio.sendValue("signal", 3)
          basic.pause(500)
      } else if (lightbutton == 1) {
          radio.sendValue("signal", 7)
          basic.pause(500)
      } else if (lightbutton == 0) {
          radio.sendValue("signal", 8)
          basic.pause(500)
      } else {
          radio.sendValue("signal", 0)
          basic.pause(500)
      }
```

```
    }
})
```

---

**Right hand**

```
let compassangle = 0
let speed = 0
let speedIO = 0
let CompassIO = 0
let signal = 0
let yacc = 0
let xacc = 0
radio.onReceivedValue(function (name, value) {
    if (name == "signal") {
        signal = value
    }
    if (name == "speed") {
        speed = value
    }
})
function compass2() {
    if (compassangle > 337 || compassangle <= 22) {
        basic.showArrow(ArrowNames.North)
    } else if (compassangle > 22 && compassangle <= 67) {
        basic.showArrow(ArrowNames.NorthWest)
    } else if (compassangle > 67 && compassangle <= 112) {
        basic.showArrow(ArrowNames.West)
    } else if (compassangle > 112 && compassangle <= 157) {
        basic.showArrow(ArrowNames.SouthWest)
    } else if (compassangle > 157 && compassangle <= 202) {
        basic.showArrow(ArrowNames.South)
    } else if (compassangle > 202 && compassangle <= 247) {
        basic.showArrow(ArrowNames.SouthEast)
```

```
    } else if (compassangle > 247 && compassangle <= 292) {
        basic.showArrow(ArrowNames.East)
    } else if (compassangle > 292 && compassangle <= 337) {
        basic.showArrow(ArrowNames.NorthEast)
    }
}
input.onButtonPressed(Button.A, function () {
    if (CompassIO == 0) {
        CompassIO = 1
        speedIO = 0
    } else if (CompassIO == 1) {
        CompassIO = 0
        basic.clearScreen()
    }
})
input.onButtonPressed(Button.B, function () {
    if (speedIO == 0) {
        speedIO = 1
        CompassIO = 0
    } else if (speedIO == 1) {
        speedIO = 0
        basic.clearScreen()
    }
})
xacc = 0
yacc = 0
radio.setGroup(17)
basic.forever(function () {
    if (signal != 0) {
        if (CompassIO == 1) {
            compassangle = input.compassHeading()
            compass2()
        }
        if (speedIO == 1) {
```

```
        basic.showNumber(speed)
    }
    xacc = input.acceleration(Dimension.X)
    yacc = input.acceleration(Dimension.Y)
    if (yacc < -900) {
        radio.sendValue("signal", 2)
        basic.pause(100)
    } else if (xacc < -900) {
        radio.sendValue("signal", 6)
        basic.pause(100)
    } else {

    }
  }
})
```

---

**On helmet**

```
let frontlight = 0
let xacc = 0
let signal = 0
radio.onReceivedValue(function (name, value) {
  if (name == "signal") {
    signal = value
    if (signal == 7) {
        frontlight = 1
    }
    if (signal == 8) {
        frontlight = 0
    }
  }
})
input.onButtonPressed(Button.A, function () {
```

```
})
radio.setGroup(17)
basic.forever(function () {
  basic.showNumber(signal)
  if (signal != 0) {
    xacc = input.acceleration(Dimension.X)
    if (xacc < -900 || xacc > 900) {
      pins.digitalWritePin(DigitalPin.P4, 1)
      music.playTone(262, music.beat(BeatFraction.Whole))
      pins.digitalWritePin(DigitalPin.P8, 1)
      pins.digitalWritePin(DigitalPin.P13, 1)
      pins.digitalWritePin(DigitalPin.P14, 1)
      pins.digitalWritePin(DigitalPin.P16, 1)
      basic.pause(500)
      pins.digitalWritePin(DigitalPin.P8, 0)
      pins.digitalWritePin(DigitalPin.P13, 0)
      pins.digitalWritePin(DigitalPin.P14, 0)
      pins.digitalWritePin(DigitalPin.P16, 0)
    } else {
      if (signal == 1) {
        pins.digitalWritePin(DigitalPin.P16, 0)
        pins.digitalWritePin(DigitalPin.P8, 1)
        basic.pause(500)
        pins.digitalWritePin(DigitalPin.P8, 0)
        basic.pause(100)
      } else if (signal == 2) {
        pins.digitalWritePin(DigitalPin.P16, 0)
        pins.digitalWritePin(DigitalPin.P13, 1)
        basic.pause(500)
        pins.digitalWritePin(DigitalPin.P13, 0)
        basic.pause(100)
      } else if (signal == 3) {
        pins.digitalWritePin(DigitalPin.P16, 1)
      } else if (signal == 4) {
```

```
            pins.digitalWritePin(DigitalPin.P16, 1)
            basic.pause(500)
            pins.digitalWritePin(DigitalPin.P16, 0)
            basic.pause(100)
        } else if (signal == 5) {

        } else if (signal == 6) {
            music.playTone(262, music.beat(BeatFraction.Whole))
            pins.digitalWritePin(DigitalPin.P8, 1)
            pins.digitalWritePin(DigitalPin.P13, 1)
            basic.pause(500)
            pins.digitalWritePin(DigitalPin.P8, 0)
            pins.digitalWritePin(DigitalPin.P13, 0)
            basic.pause(100)
        } else if (signal == 7) {
            pins.digitalWritePin(DigitalPin.P16, 0)
            pins.digitalWritePin(DigitalPin.P14, 1)
        } else if (signal == 8) {
            pins.digitalWritePin(DigitalPin.P16, 0)
            pins.digitalWritePin(DigitalPin.P14, 0)
            pins.digitalWritePin(DigitalPin.P4, 1)
        }
    }
  } else {
    pins.digitalWritePin(DigitalPin.P0, 0)
    pins.digitalWritePin(DigitalPin.P8, 0)
    pins.digitalWritePin(DigitalPin.P13, 0)
    pins.digitalWritePin(DigitalPin.P14, 0)
    pins.digitalWritePin(DigitalPin.P15, 0)
    pins.digitalWritePin(DigitalPin.P16, 0)
    pins.digitalWritePin(DigitalPin.P4, 1)
  }
})
------------------------
```

**On wheel**

```
let check = 0
let checkf = 0
let checke = 0
let checkd = 0
let checkc = 0
let checkb = 0
let checka = 0
let speed = 0
let adjspeedj = 0
let adjspeedi = 0
let adjspeedh = 0
let adjspeedg = 0
let adjspeedf = 0
let adjspeede = 0
let adjspeedd = 0
let H = 0
let G = 0
let adjsppedc = 0
let F = 0
let E = 0
let adjspeedb = 0
let D = 0
let C = 0
let adjspeeda = 0
let B = 0
let A = 0
radio.setGroup(17)
basic.forever(function () {
    A = input.acceleration(Dimension.Y)
    basic.pause(5)
    B = input.acceleration(Dimension.Y)
    basic.pause(5)
    adjspeeda = Math.round(Math.abs(B - A) * 0.09 / (2 * B * 0.001) * 3.6)
```

```
        basic.pause(5)
        C = input.acceleration(Dimension.Y)
        basic.pause(5)
        D = input.acceleration(Dimension.Y)
        basic.pause(5)
        adjspeedb = Math.round(Math.abs(D - C) * 0.09 / (2 * D * 0.001) * 3.6)
        basic.pause(5)
        E = input.acceleration(Dimension.Y)
        basic.pause(5)
        F = input.acceleration(Dimension.Y)
        basic.pause(5)
        adjsppedc = Math.round(Math.abs(F - E) * 0.09 / (2 * F * 0.001) * 3.6)
        basic.pause(5)
        G = input.acceleration(Dimension.Y)
        basic.pause(5)
        H = input.acceleration(Dimension.Y)
        basic.pause(5)
        adjspeedd = Math.round(Math.abs(H - G) * 0.09 / (2 * H * 0.001) * 3.6)
        basic.pause(5)
        adjspeede = Math.round(Math.abs(H - A) * 0.09 / (2 * H * 0.05) * 3.6)
        adjspeedf = Math.round(Math.abs(H - C) * 0.09 / (2 * H * 0.04) * 3.6)
        adjspeedg = Math.round(Math.abs(H - E) * 0.09 / (2 * H * 0.02) * 3.6)
        adjspeedh = Math.round(Math.abs(F - B) * 0.09 / (2 * F * 0.03) * 3.6)
        adjspeedi = Math.round(Math.abs(F - D) * 0.09 / (2 * F * 0.01) * 3.6)
        adjspeedj = Math.round(Math.abs(D - A) * 0.09 / (2 * D * 0.03) * 3.6)
        speed = Math.abs(Math.round((adjspeeda + (adjspeedb + (adjsppedc + (adjspeedd +
(adjspeede + (adjspeedf + (adjspeedg + (adjspeedh + (adjspeedi + adjspeedj)))))))) / 10))
        checka = 0.005 * (adjsppedc - adjspeedb) + 0.005 * (adjspeedb - adjspeeda)
        checkb = 0.005 * (adjspeedd - adjsppedc) + 0.005 * (adjsppedc - adjspeedb)
        checkc = 0.005 * (adjspeede - adjspeedd) + 0.005 * (adjspeedd - adjsppedc)
        checkd = 0.005 * (adjspeedf - adjspeede) + 0.005 * (adjspeede - adjspeedd)
        checke = 0.005 * (adjspeedg - adjspeedf) + 0.005 * (adjspeedf - adjspeede)
        checkf = 0.005 * (adjspeedh - adjspeedg) + 0.005 * (adjspeedg - adjspeedf)
        check = (checka + checkb + checkc + checkd + checke + checkf) / 6
```

```
        if (check * 0.9 >= speed) {
        if (check * 1.1 <= speed) {
        radio.sendValue("speed", speed)
        basic.clearScreen()
        }
        }
})
```